Touch–Free Hci Using Cnn

Sivaranjani S[1], Vasundra L[2], Yeswanth E[3], Iniya Akash R[4], J Priyadharshini[5]
[1,2,3,4,5]*Department of Artificial Intelligence and Data Science.Salem college of Engineering and Technology, Tamil nadu, India.*

***Abstract*—A growing world need touchless and smart ways to interact with computers. Traditional input devices like the mouse and keyboard require physical contact, which is not suitable for hygienic places, smart environments, or for people who need assistive technologies. To solve this problem, this project introduces a Touch free Real-time virtual Using Hand Gestures, developed using CNN in Python.**

The system focuses on improving usability, flexibility, long-distance interaction, and security. The proposed system allows users to control mouse functions such as moving the cursor, clicking, dragging, long pressing, scrolling, volume levels, and multiple accessing files using simple hand gestures. A camera captures live video of the user's hand, and the video is processed continuously to detect and recognize gestures. Instead of depending on a fixed number of fingers, the system understands gestures based on hand shape and movement.

It can also work for users with flexible hands, bent fingers, or even six fingers. One of the main highlights of this project is its ability to support hand gesture control from a long distance. With the help of high-quality or zoom-enabled cameras, the system is designed to work from a distance of about 30 to 40 feet. The system automatically adjusts cursor movement speed and sensitivity based on how far the user is from the camera, ensuring smooth and accurate control. The project also includes basic security features to protect the system and user data.

The camera checks the background before allowing gesture control, and unauthorized access can block the system from working. File access through gestures is controlled, and no camera video is stored, which helps protect user privacy. Overall, this virtual mouse system provides a simple, safe, and touchless way to interact with computers.

It can be used in places like smart classrooms, offices, hospitals, assistive technology systems, and secure work areas. The system can also be improved further in the future with better hardware, support for multiple users, and stronger security features.

***Index Terms*—Touch–Free Interaction, HCI, Media pipe, CNN, Gestures**

I. INTRODUCTION

Human–Computer Interaction (HCI) focuses on improving the way users interact with digital systems. Traditional input devices such as the mouse and keyboard require physical contact, which can create hygiene, comfort, and accessibility issues, particularly highlighted during situations like the COVID-19 pandemic. To overcome these limitations, AI-based virtual mouse systems have been introduced as a touch-free alternative

Virtual mouse systems use computer vision and artificial intelligence to interpret hand gestures captured through a camera and convert them into mouse actions. Advances in frameworks such as OpenCV and MediaPipe enable accurate real-time hand landmark detection and finger tracking, allowing smooth and reliable cursor control, clicking, and scrolling without additional hardware.

These systems reduce hardware dependency, improve accessibility for users with physical disabilities, and support hygienic interaction in shared environments. This project focuses on developing a real-time AI-based virtual mouse system that achieves accurate gesture recognition and smooth cursor control using modern computer vision techniques, contributing to more natural and accessible human– computer interaction.

II. LITERATURE REVIEW SUMMARY

Human–Computer Interaction (HCI) has gained significant attention with the widespread use of computers in daily life. Traditional input devices such as the mouse and keyboard require physical contact, which raises concerns related to hygiene, comfort, and

accessibility—especially in healthcare settings, public environments, and during situations like the COVID-19 pandemic. To address these challenges, researchers have increasingly explored AI-based virtual mouse systems that enable touch-free interaction using hand gestures.

Early virtual mouse systems relied on wearable devices such as data gloves, which offered high accuracy but were expensive, uncomfortable, and impractical for everyday use. This led to the adoption of vision-based approaches using webcams and image processing techniques. Initial methods focused on skin color detection, motion analysis, face tracking, and color-based markers. Although these systems were cost-effective and simple to implement, their performance was highly sensitive to lighting conditions, background complexity, and camera quality. Additionally, marker-based systems required users to wear colored caps or tapes, reducing usability and reliability in real-world environments.

Subsequent research emphasized hand gesture recognition using machine learning and computer vision techniques. Convolutional Neural Networks (CNNs) were introduced to improve gesture recognition accuracy; however, their high computational requirements limited real-time performance on low-end systems. To overcome these limitations, researchers increasingly adopted lightweight and efficient frameworks such as Media Pipe and OpenCV. Media Pipe Hands enables real-time detection of 21 hand landmarks using a standard webcam, allowing precise tracking of finger movements without external hardware. OpenCV supports real-time video capture and image processing, while tools like PyAutoGUI are used to simulate mouse actions such as cursor movement, clicking, dragging, and scrolling.

Recent studies demonstrate that finger- tracking-based virtual mouse systems provide improved accuracy, responsiveness, and user comfort compared to earlier approaches. Gesture- based mappings, such as using the index finger for cursor movement and specific finger combinations for clicking or scrolling, offer intuitive and natural interaction. Smoothing techniques further enhance stability by reducing cursor jitter. These systems have shown high accuracy and are particularly beneficial for assistive technology, hygiene- sensitive environments, smart classrooms, and public kiosks.

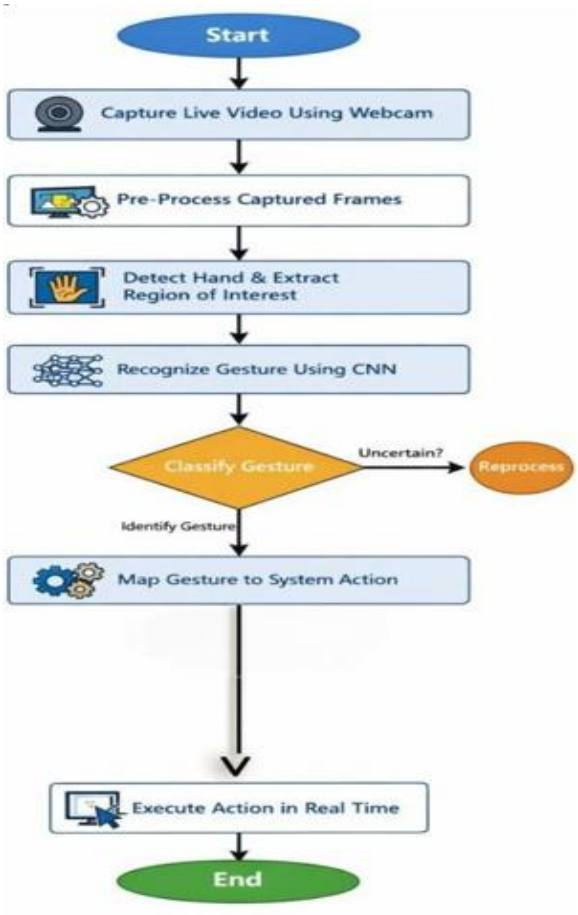
Despite significant progress, challenges remain. System performance can degrade under poor lighting conditions, rapid hand movements, or limited hardware resources. Complex gestures are still more difficult to recognize reliably, and calibration is often required for consistent performance. Nevertheless, the literature clearly indicates that AI-based virtual mouse systems are a promising alternative to traditional input
devices. They offer touch-free, low-cost, and accessible interaction, forming a strong foundation for the development of more robust, accurate, and user- friendly real-time virtual mouse systems.

III. METHODOLOGY

The proposed virtual mouse system operates using real-time video captured through a camera. The live video stream is divided into individual frames and preprocessed using OpenCV image processing algorithms, including Gaussian Blur for noise reduction and adaptive thresholding for lighting normalization.
Hand detection and feature extraction are performed using the Media Pipe Hand Landmark Detection algorithm, which identifies 21 key hand landmarks. Gesture recognition is achieved by applying geometric feature analysis, where relative distances and angles between landmarks are computed to classify gestures without relying on fixed finger counting, enabling support for flexible and non-standard hand structures.
For long-distance interaction, a distance estimation and gesture scaling algorithm is applied using palm size and perspective normalization to dynamically adjust cursor sensitivity. Smooth cursor movement is ensured using a Kalman Filter algorithm to predict hand motion and reduce jitter.

Recognized gestures are mapped to mouse actions using gesture-to-action mapping logic, and mouse control is executed through system- level input simulation. Basic camera background validation is implemented using background subtraction techniques to restrict unauthorized access. All

processing is performed in real time without storing video frames, ensuring secure and confidential data handling.



Start

The system begins when the program is launched. All required libraries and the camera are initialized.

Capture Live Video Using Webcam

- The webcam starts capturing live video continuously.
- The system receives video in the form of frames (images captured every second).

Pre-Process Captured Frames

The captured frames are processed before analysis. This step improves accuracy and performance.

- Pre-processing may include:
- Resizing the frame
- Flipping the image for mirror view
- Adjusting brightness and contrast
- Removing noise

This makes the image clearer and easier for the system to analyze.

Detect Hand

- The system detects the hand in the video frame.
- Once the hand is detected:
- Only the hand portion is extracted
- Background is ignored
- The system focuses only on the important area (Region of Interest)
- This reduces unnecessary data and increases speed.

Recognize Gesture Using CNN

The extracted hand image is given to a CNN (Convolutional Neural Network).

- Analyzes finger positions
- Identifies patterns
- Predicts the type of hand gesture
- It compares the input with trained data to recognize the gesture correctly.

Classify Gesture

The system classifies the gesture based on CNN output.

- If the prediction confidence is low
- The system reprocesses the frame.
- If the prediction is clear:
- The gesture is confirmed and sent to the next step.

Map Gesture to System Action

Each recognized gesture is linked to a specific computer action.

For example:

- One finger → Move cursor
- Two fingers → Left click
- Three fingers → Right click
- Pinch gesture → Scroll

The system converts the gesture into a command.

Execute Action in Real Time

The mapped action is performed immediately on the computer screen.

For example:

- Cursor moves
- Click happens

- Scrolling occurs

All actions happen in real time without delay.
End
The process continues in a loop until the program is stopped.

IV. SYSTEM REQUIREMENTS

The system requires a computer with at least an Intel i5 or i7 processor (8th generation or above), 8 GB of RAM, and around 10 GB of SSD storage to ensure smooth and fast processing. An HD webcam with 1080p resolution is needed for capturing hand gestures in real time. An NVIDIA GPU is optional but useful if deep learning frameworks such as PyTorch are used for faster computation. The software environment should be either Windows 11 or Ubuntu 22.04, with Python version between 3.9 and 3.11. The main libraries used in this project are OpenCV and NumPy, along with built-in Python modules such as Time and OS. For development, tools like Visual Studio Code or PyCharm can be used, and Git may be used for version control if required. Functionally, the system is designed to perform real-time hand detection, recognize gestures, control the cursor, and support click and drag operations, with optional segmentation for improved accuracy.

V. SIMULATION / PACKAGES

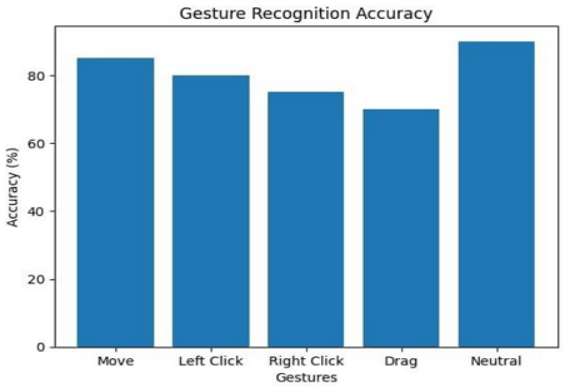
The system is simulated using a real-time webcam feed to create a virtual touch mouse environment. The webcam continuously captures video frames, which are processed to detect the hand and recognize gestures without noticeable delay. The frames are flipped horizontally to provide a mirror view so that user movements feel natural. The system detects the hand region, extracts key landmarks such as fingertips and joints, and uses their relative positions to identify gestures like cursor movement, click, and drag. The hand position is mapped to screen coordinates to control the mouse pointer in real time.
The system was tested under different conditions. It performs well in normal lighting with stable detection and smooth gesture recognition. In low-light environments, accuracy slightly decreases because landmark detection becomes less precise. Complex backgrounds may cause minor detection errors, but overall performance remains acceptable. The best results are obtained when the hand is kept at a distance of about 30–80 cm from the camera. When multiple hands appear, the system processes only one hand to avoid confusion. During continuous use, the system remains stable, although slight latency may occur depending on hardware.

The Implementation is done in Python using OpenCV for video capture and frame processing, and NumPy for mathematical operations and cursor mapping. Hydra and OmegaConf are used for managing configuration settings, and PyTorch is optionally used when segmentation is enabled to improve hand isolation. The system is structured into custom modules, including a camera module for video input, a hand detector for landmark extraction, a segmenter for background separation, a tracker for smoothing cursor movement, and a mouse controller for converting gestures into system actions.

In terms of performance, the system achieves around 10–20 frames per second, which is sufficient for real-time interaction. Latency is generally low, though slightly higher on low-end systems. Basic gestures are detected quickly, and cursor movement appears smooth due to filtering techniques that reduce noise and jitter. Accuracy is high in controlled environments but slightly lower in challenging conditions such as poor lighting. The system uses moderate CPU resources, while enabling deep learning features increases computational load.

VI. RESULTS AND DISCUSSION

The touchless mouse control system is about 80% complete and works in real time using a webcam with an average speed of 20–25 FPS. It can perform basic operations such as cursor movement, left click, right click, and drag using hand landmarks. Cursor movement is the most accurate, while right click and drag need improvement due to similar finger patterns and sensitivity to hand angle.



The system performs well in good lighting and simple backgrounds but loses some accuracy in complex environments. The One Euro Filter helps reduce cursor jitter and makes movement smoother, though small fluctuations still occur during fast motion. Hand-to- screen mapping works effectively, but slight errors appear near screen edges. An optional segmentation module improves hand visibility but increases computational load and is not fully optimized. The modular design allows future upgrades such as deep learning–based gesture recognition, better calibration, and multi-hand tracking. Overall, the system achieves its main goal of touchless mouse control but requires further optimization for real-world use.